UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/782,405	02/12/2001	Clemente Izurieta	10006526-1	1881

22879   7590   08/02/2005

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER
KANG, INSUN

ART UNIT ·	PAPER NUMBER
2193	

DATE MAILED: 08/02/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

	Application No.	Applicant(s)
Office Action Summary	09/782,405	IZURIETA, CLEMENTE
	Examiner	Art Unit	
	Insun Kang	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE <u>3</u> MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on <u>28 April 2005</u>.

2a)☒ This action is **FINAL**.   2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) <u>1,4,6,7,9,10 and 24-28</u> is/are pending in the application.

    4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) <u>1, 4, 6, 7, 9, 10, 24-28</u> is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

    Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

    Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

    a)☐ All   b)☐ Some * c)☐ None of:

      1.☐ Certified copies of the priority documents have been received.

      2.☐ Certified copies of the priority documents have been received in Application No. _____.

      3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

    * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date _____.

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1.   This action is in response to the amendment filed 4/28/2005.

2.   As per applicant's request, claims 1, 4, 6, and 24 have been amended. Claims 1,

4, 6, 7, 9, 10, 24-28 are pending in the application.

Claim Objections

3.   The objection to claim 6 has been withdrawn due to the amendment to the claim.

Claim Rejections - 35 USC § 112

4.   The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly
> claiming the subject matter which the applicant regards as his invention.

Claims 1, 4, 6, 7, 9, 10, 24-28 are rejected under 35 U.S.C. 112, second

paragraph, as being indefinite for failing to particularly point out and distinctly claim the

subject matter which applicant regards as the invention.

Per claims 1, 4, and 24, the limitation, "bi-directional access" is unclear as to how such

access to the object oriented data from the non-object oriented data is performed. It is

interpreted as: the mapping techniques to the non-object data can "potentially" allow to

map the non-object data to the object oriented data where the techniques can be type

casts, wrappers, and/or inheritance mechanism.

As per claims 6, 7, 9, 10, and 25-28, these claims are rejected for dependency

on the above rejected parent claims.

Claim Rejections - 35 USC § 102

5.   The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that

form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6.   Claims 1, 3-6, and 9-11 are rejected under 35 U.S.C. 102(b) as being anticipated

by Applicant's Admitted Prior Art (hereinafter referred to as "APA") disclosed in the

instant application.

Per claim 4:

APA teaches:

-loading memory with non-object oriented data ("there are several techniques that use

zero-size objects to achieve a mapping between non-object oriented data models and

object oriented based models. Zero-sized object mapping is a form of overlying existing

memory occupied by another object. Some prior coding techniques to achieve zero-

size object mapping are reinterpret casts," APA, page 2; "legacy infrastructures...CAD,"

page 1)

- mapping an object oriented model to an image in memory space occupied by the non-

object oriented data without requiring additional memory space ("there are several

techniques that use zero-size objects to achieve a mapping between non-object

oriented data models and object oriented based models," APA, page 2; "if the memory

images of the types are different," page 2)

-retrieving a non-object oriented data element from the memory in the object oriented

model based on the mapping ("there are several techniques that use zero-size objects

to achieve a mapping between non-object oriented data models and object oriented

based models. Zero-sized object mapping is a form of overlying existing memory

occupied by another object. Some prior coding techniques to achieve zero-size object

mapping are reinterpret casts," APA, page 2)

-wherein the object oriented data inherits the non-object oriented data ("Another

concept in object oriented programming is inheritance. Inheritance is the ability to

derive a new class from one or more existing classes. The new class, known as a

subclass, may inherit or incorporate all properties of a base class, including its attributes

and its methods," page 4 lines 19-22)

-providing bi-directional access, wherein the bi-directional access permits retrieval of the

object oriented data based on the non-object oriented data ("there are several

techniques that use zero-size objects to achieve a mapping between non-object

oriented data models and object oriented based models. Zero-sized object mapping is

a form of overlying existing memory occupied by another object. Some prior coding

techniques to achieve zero-size object mapping are reinterpret casts," APA, page 2) as

claimed.

Per claim 6:

Inheritance is one of important concepts in object-oriented programming (OOP).

APA recites, "Another concept in object oriented programming is inheritance.

Inheritance is the ability to derive a new class from one or more existing classes. The

new class, known as a subclass, may inherit or incorporate all properties of a base

class, including its attributes and its methods (page 4 lines 19-22)." Also, the applicant

states, "When a programmer creates an instance of that structure in memory, the

programmer has to allocate memory to hold that information. In order to map, the

object oriented system inherits the non-object oriented data that came from the C

structure. Inheritance allows the programmer to access the non-object oriented

structure or any other base structure's data (page 5 lines 9-10)." Non-object oriented

data cannot be directly inherited because non-object oriented data structure does not

have the concept of class mechanism. The specification does not describe how this

inheritance used "to access the non-object oriented structure or any other base

structure's data" and to create "child class A 210" that represents the "definition of the

object oriented based object A 205 and drives it from the non-object oriented C

structure 200" works differently from the conventional OOP inheritance mechanism that

uses a base wrapper class to access legacy data. Therefore, the examiner considers

that the inheritance mechanism stated in the instant specification (page 5 lines 7-14) is

the conventional inheritance mechanism. Accordingly, APA teaches deriving a class

from the non-object oriented data (page 5 lines 7-14) as claimed.

Per claim 9:

The rejection of claim 4 is incorporated, and further, Inheritance is one of important

concepts in object-oriented programming (OOP). APA recites, "Another concept in

object oriented programming is inheritance. Inheritance is the ability to derive a new

class from one or more existing classes. The new class, known as a subclass, may
inherit or incorporate all properties of a base class, including its attributes and its
methods (page 4 lines 19-22)." Also, the applicant states, "When a programmer creates
an instance of that structure in memory, the programmer has to allocate memory to hold
that information. In order to map, the object oriented system inherits the non-object
oriented data that came from the C structure. Inheritance allows the programmer to
access the non-object oriented structure or any other base structure's data (page 5 lines
9-10)." Non-object oriented data cannot be directly inherited because non-object
oriented data structure does not have the concept of class mechanism. The
specification does not describe how this inheritance used "to access the non-object
oriented structure or any other base structure's data" and to create "child class A 210"
that represents the "definition of the object oriented based object A 205 and drives it
from the non-object oriented C structure 200" works differently from the conventional
OOP inheritance mechanism that uses a base wrapper class to access legacy data.
Therefore, the examiner considers that the inheritance mechanism stated in the instant
specification (page 5 lines 7-14) is the conventional inheritance mechanism.
Accordingly, APA teaches accessing the non-object oriented data using the object
oriented model as claimed.

Per claim 10:

The rejection of claim 4 is incorporated, and further, APA teaches retrieving occurs with
zero size memory ("there are several techniques that use zero-size objects to achieve a

mapping between non-object oriented data models and object oriented based models.

Zero-sized object mapping is a form of overlying existing memory occupied by another

object. Some prior coding techniques to achieve zero-size object mapping are

reinterpret casts," APA, page 2) as claimed.

Per claim 1, it is the apparatus version of claim 4, respectively, and is rejected for

the same reasons set forth in connection with the rejection of claim 4 above.

Claim Rejections - 35 USC § 103

7.   The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set
> forth in section 102 of this title, if the differences between the subject matter sought to be patented and
> the prior art are such that the subject matter as a whole would have been obvious at the time the
> invention was made to a person having ordinary skill in the art to which said subject matter pertains.
> Patentability shall not be negatived by the manner in which the invention was made.

8.   Claim 7 is rejected under 35 U.S.C. 103(a) as being unpatentable over

Applicant's Admitted Prior Art (hereinafter referred to as "APA") disclosed in the instant

application in view of Wise (Casting in C++: "Brining Safety and Smartness to Your

Programs," Coding-zone 1996).

Per claim 7:

The rejection of claim 6 is incorporated, and further, APA discloses a reinterpret_cast

but does not explicitly teach instantiating an instance of the class based on static

casting. However, Wise teaches that C++ offers several different ways to cast an

expression to a different type and static casting was a known programming language

feature in the art of computer software development at the time applicant invention was made to "convert a base class pointer to a derived class pointer... (page 2, The static_cast Operator)" and to "perform the explicit inverse of the implicit standard conversions (page 3)." It would have been obvious for one having ordinary skill in the art of software development to modify APA's disclosed method to use static casting as the reinterpret casting result is "usually implementation dependent and, therefore, not likely to be portable (Wise, page 7, The reinterpret_cast Operator)"and the conversion "between totally unrelated types of objects can "cause serious problems if the memory images of the types are different (APA, page 2 lines 9-14)." The modification would be obvious because one having ordinary skill in the art would be motivated to instantiate an object through static casting because it can be used to perform conversions explicitly defined in classes, to cast a pointer of a derived class to its base class implicitly, and to reverse any implicit type conversion (Wise, page 3) performing compile time type checks with certain restrictions as taught by Wise.

9.   Claims 1, 4, 6, 7, 9, 10, and 24-27 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 6,305,007 to Mintz in view of Wise (Casting in C++: "Brining Safety and Smartness to Your Programs," Coding-zone 1996).

Per claim 4:

Mintz teaches loading memory with non-object oriented data ("The new property pages interact with the legacy data structures via classes which descend from the base wrapper," col 4 lines 1-34).

Mintz discloses deriving a class from a base class that wraps legacy data structure, as non-object oriented data cannot be directly inherited because non-object oriented data structure does not have the concept of class mechanism. Mintz does not explicitly teach mapping an object oriented model onto a memory space occupied by the non-object oriented data without requiring additional memory space. However, Wise teaches that C++ offers several different ways to cast an expression to a different type such as reinterpret and static casting and specifically static casting was known in the art of computer software development at the time applicant invention was made to "convert a base class pointer to a derived class pointer... (page 2, The static_cast Operator)" and to "perform the explicit inverse of the implicit standard conversions (page 3)." It would have been obvious for one having ordinary skill in the art of software development to modify Mintz's disclosed method to use the static casting mechanism to access the non-object oriented data from the base class without allocating a new memory.

The modification would be obvious because one having ordinary skill in the art would be motivated to access the legacy data using the object oriented concepts such as encapsulation and abstraction overlaying the memory where the legacy data occupy by static casting performing compile time type checks with certain restrictions (Wise, page 3) as taught by Wise.

Mintz further teaches retrieving a non-object oriented data element from the memory in the object oriented model based on the mapping wherein the object oriented data inherits the non-object oriented data ("The new property pages interact with the legacy data structures via classes which descend from the base wrapper," col 4 lines 1-34; "This class is inherited from the base wrapper class (so as to be able to access the property page framework) and the policy class associated with each property that needs manipulation," col 4 lines 30-34; see also col 6 lines 22-31; col 4 lines 60-67);

As shown above, Mintz in view of Wise discloses accessing the legacy data through inheritance and casting. Therefore Mintz discloses providing bi-directional access, wherein the bi-directional access permits retrieval of the object oriented data based on the non-object oriented as claimed.

Per claim 6:

The rejection of claim 5 is incorporated, and further, Mintz teaches deriving a class from the non-object oriented data ("The new property pages interact with the legacy data structures via classes which descend from the base wrapper," col 4 lines 1-34)

Per claim 7:

The rejection of claim 6 is incorporated, and further, Mintz does not explicitly teach instantiating an instance of the class based on static casting. However, Wise teaches that C++ offers several different ways to cast an expression to a different type such as reinterpret and static casting and specifically static casting was known in the art

of computer software development at the time applicant invention was made to "convert

a base class pointer to a derived class pointer... (page 2, The static_cast Operator)"

and to "perform the explicit inverse of the implicit standard conversions (page 3)." It

would have been obvious for one having ordinary skill in the art of software

development to modify Mintz's disclosed method to use static casting mechanism to

access the non-object oriented data from the base class without allocating a new

memory.

The modification would be obvious because one having ordinary skill in the art

would be motivated to access the legacy data using the object oriented concepts such

as encapsulation and abstraction overlaying the memory where the legacy data occupy

by static casting performing compile time type checks with certain restrictions (Wise,

page 3) as taught by Wise.

Per claim 9:

The rejection of claim 4 is incorporated, and further, Mintz teaches accessing the

non-object oriented data using the object oriented model (col 3 lines 15-35; ("The new

property pages interact with the legacy data structures via classes which descend from

the base wrapper," col 4 lines 1-34; "This class is inherited from the base wrapper class

(so as to be able to access the property page framework) and the policy class

associated with each property that needs manipulation," col 4 lines 30-34; see also col 6

lines 22-31; col 4 lines 60-67).

Per claim 10:

The rejection of claim 4 is incorporated, and further, this claim is another version of the

claimed method discussed in claim 6 above, wherein all claim limitations also have

been addressed and/or covered in cited areas as set forth the above. Therefore,

accordingly, Mintz teaches retrieving occurs with zero size memory as claimed.

Per claim 1, it is the apparatus version of claim 4, respectively, and is rejected for

the same reasons set forth in connection with the rejection of claim 4 above.

Per claims 24-27, they are another method versions of claims 4, 6,7,9, and 10

except the further limitation, "legacy C-structure data." However, Mintz specifically

deals with C as the legacy data and C++ as the object model ("a legacy

program...written in C," col. 3 lines 25-35; "a wrapper object written in an object-

oriented language such as C++," col. 3 lines 35-46), respectively, and are rejected for

the same reasons set forth in connection with the rejection of claims 4, 6,7,9, and 10

above.

10.   Claim 28 is rejected under 35 U.S.C. 103(a) as being unpatentable over U.S.

Patent 6,305,007 to Mintz, in view of Wise (Casting in C++: "Brining Safety and

Smartness to Your Programs," Coding-zone 1996), and further in view of Applicant's

Admitted Prior Art (hereinafter referred to as "APA") disclosed in the instant application.

Per claim 28:

Mintz and Wise do not explicitly teach the non-object oriented data and the object

oriented data is associated with very large scale integrated circuits. However, most

modern chips employ VLSI architectures (or ULSI). APA specifically teaches VLSI was

known in the art of computer architecture, at the time applicant's invention was made, to

integrate many individual functions or systems by combining numerous transistors into a

single chip. It would have been obvious for one having ordinary skill in the art of

computer architecture to modify the system of Mintz and Wise to incorporate the

teachings of APA. The modification would be obvious because one having ordinary skill

in the art would be motivated to integrate many object oriented or non-object oriented

individual functions or systems by combining numerous transistors into a single chip as

indicated by APA (page 1).

11.   Claims 24-27 are rejected under 35 U.S.C. 103(a) as being unpatentable over

U.S. Patent 5,499,371 to Henninger et al., hereinafter referred to as "Henninger," in

view of Wise (Casting in C++: "Brining Safety and Smartness to Your Programs,"

Coding-zone 1996).

Per claim 24:

Henninger teaches:

-mapping a data object representing object oriented data to an image in memory

occupied by the non-object oriented data represented ("automatically map information

between an object-oriented application and a structured database, such as a relational

database," col. 2 lines 55-67). Henninger does not explicitly teach that the non-object

data is legacy C-structure data. However, Henninger specifically states that the

underlying database or its structure does not need to be known in order to be accessed

(col. 3 lines 1-25). Therefore, it is obvious that any legacy code such as C data can be

manipulated in Henninger's system.

-creating a child class based on inheriting the legacy C-structure data by the data object

("Each routine takes into account the full semantics of the object model,

including...inheritances...in the object model," col. 7 lines 30-41)

Henninger does not explicitly teach accessing the child class by static casting the

non-object oriented data represented by the legacy C-structure data with the object

oriented data. However, Wise teaches that C++ offers several different ways to cast an

expression to a different type such as reinterpret and static casting and specifically

static casting was known in the art of computer software development at the time

applicant invention was made to "convert a base class pointer to a derived class

pointer... (page 2, The static_cast Operator)" and to "perform the explicit inverse of the

implicit standard conversions (page 3)." It would have been obvious for one having

ordinary skill in the art of software development to modify Henninger's disclosed method

to use the static casting mechanism to access the non-object oriented data from the

base class without allocating a new memory. The modification would be obvious

because one having ordinary skill in the art would be motivated to access the legacy

data using the object oriented concepts such as encapsulation and abstraction

overlaying the memory where the legacy data occupy by static casting performing

compile time type checks with certain restrictions (Wise, page 3) as taught by Wise.

As shown above, Henninger in view of Wise discloses accessing the legacy data

through inheritance and static casting, therefore, Henninger discloses providing bi-

directional access, wherein the bi-directional access permits retrieval of the object

oriented data based on the non-object oriented data as claimed.

Per claim 25:

The rejection of claim 24 is incorporated, and further, Henninger teaches:

Using zero size mapping is used to map the data object representing object oriented

data to the image in memory occupied by the non-object oriented data ("automatically

map information between an object-oriented application and a structured database,

such as a relational database," col. 2 lines 55-67).

Per claim 26:

The rejection of claim 25 is incorporated, and further, Henninger teaches deriving object

oriented classes from the legacy C-structures at compile time (i.e. col. 6 lines 7-15).

Per claim 27:

The rejection of claim 26 is incorporated, and further, Wise teaches loading memory

with non-object oriented data into the derived object oriented classes (page 2, The

static_cast Operator).

12.   Claim 28 isrejected under 35 U.S.C. 103(a) as being unpatentable over U.S.

Patent 5,499,371 to Henninger et al., hereinafter referred to as "Henninger," in view of

Wise (Casting in C++: "Brining Safety and Smartness to Your Programs," Coding-zone

1996), and further in view of Applicant's Admitted Prior Art (hereinafter referred to as

"APA") disclosed in the instant application.

Per claim 28:

Henninger and Wise do not explicitly teach the non-object oriented data and the object

oriented data is associated with very large scale integrated circuits. However, most

modern chips employ VLSI architectures (or ULSI). APA specifically teaches VLSI was

known in the art of computer architecture, at the time applicant's invention was made, to

integrate many individual functions or systems by combining numerous transistors into a

single chip. It would have been obvious for one having ordinary skill in the art of

computer architecture to modify the system of Henninger and Wise to incorporate the

teachings of APA. The modification would be obvious because one having ordinary skill

in the art would be motivated to integrate many object oriented or non-object oriented

individual functions or systems by combining numerous transistors into a single chip as

indicated by APA (page 1).

Response to Arguments

13.   Applicant's arguments filed 4/28/2005 have been fully considered but they are

not persuasive.

Per claims 1, 4 and 24:

The Applicant states that the prior art references do not teach bidirectional access
between the object and the non-object oriented data.

In response, the specification states, "using the present invention to achieve
zero-sized objects to merge the two data models allows the designers of two CAD
infrastructures a portal via which they can now potentially have bi-directional access
between models," in page 3 lines 7-11. The invention "uses" the well-known
techniques, type casting such as static_cast and inheritance to achieve the "zero-sized
objects" and to "potentially have bi-directional access." The specification mentions only
the potential of bidirectional access by such known techniques and does not further
disclose how such bidirectional access is performed in a distinct way and/or how to
retrieve the object-oriented data from the non-object oriented data specifically.
Accordingly, the examiner considers that the "zero-sized objects" mapping mechanisms
stated above permit the bi-directional access. Therefore, as shown above, the
references teach the mapping techniques between the non-object oriented data and the
object-oriented data by using the methods such as casting, inheritance, and/or wrapper.
Therefore, the rejection of the claims is considered proper and maintained.

Per claims 6-7, 9-10, and 25-28:

The applicant states that claims 6-7, 9-10, and 25-28 are allowable as being dependent
on allowable base claims. As shown above, the rejections of the independent claims
are considered proper, the argument that claims 6-7, 9-10, and 25-28 are allowable as

being dependent on an allowable base claim is considered moot. Accordingly, the rejections of claims 6-7, 9-10, and 25-28 are proper and maintained.

Conclusion

14.   **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

15.   Any inquiry concerning this communication or earlier communications from the examiner should be directed to Insun Kang whose telephone number is 571-272-3724. The examiner can normally be reached on M-F 7:30-4 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on 571-272-3719. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the

Patent Application Information Retrieval (PAIR) system. Status information for

published applications may be obtained from either Private PAIR or Public PAIR.

Status information for unpublished applications is available through Private PAIR only.

For more information about the PAIR system, see http://pair-direct.uspto.gov. Should

you have questions on access to the Private PAIR system, contact the Electronic

Business Center (EBC) at 866-217-9197 (toll-free).

Any inquiry of a general nature or relating to the status of this application should

be directed to the TC 2100 Group receptionist: 571-272-2100.

I. Kang
AU2193

ANIL KHATRI
PRIMARY EXAMINER